SHLOK IYER 1BM22CS260 SEM:3 SEC:CE

WAP to Implement doubly link list with primitive operations

I.Create a doubly linked list.

II. Insert a new node to the left of the node.

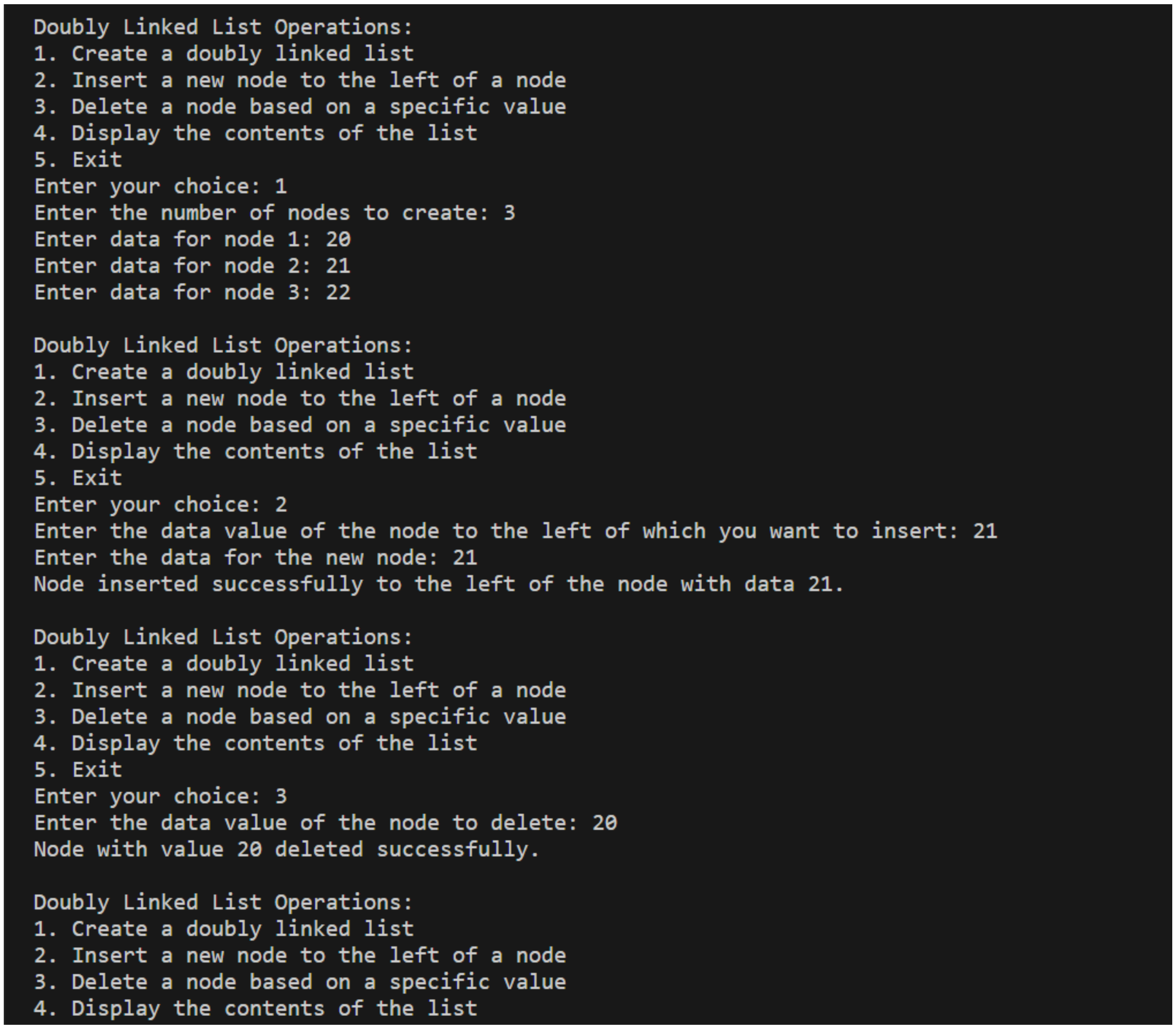
III. Delete the node based on a specific value

IV. Display the contents of the list

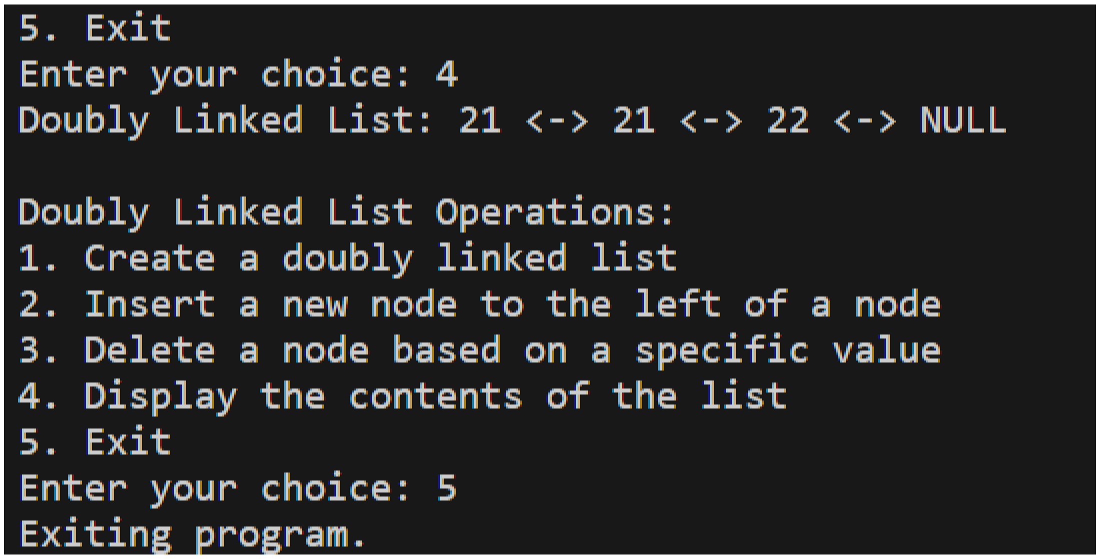
```
#include <stdio.h>
#include <stdlib.h>

// Node structure for doubly linked list
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a new node to the left of the given node
void insertLeft(struct Node** head, struct Node* target, int data) {
    if (target == NULL) {
        printf("Cannot insert to the left of a NULL target.\n");
        return;
    }

    struct Node* newNode = createNode(data);

    newNode->prev = target->prev;
    newNode->next = target;
```

```c
    if (target->prev != NULL) {
        target->prev->next = newNode;
    } else {
        *head = newNode;  // If target is the head, update head
    }

    target->prev = newNode;
}

// Function to delete the node with a specific value
void deleteNode(struct Node** head, int key) {
    struct Node* current = *head;

    while (current != NULL) {
        if (current->data == key) {
            if (current->prev != NULL) {
                current->prev->next = current->next;
            } else {
                *head = current->next;  // If the node to delete is the head
            }

            if (current->next != NULL) {
                current->next->prev = current->prev;
            }

            free(current);
            printf("Node with value %d deleted successfully.\n", key);
            return;
        }

        current = current->next;
    }

    printf("Node with value %d not found.\n", key);
}

// Function to display the contents of the doubly linked list
void display(struct Node* head) {
    struct Node* current = head;

    printf("Doubly Linked List: ");
    while (current != NULL) {
        printf("%d <-> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}


int main() {
    struct Node* head = NULL;
    int choice, data, key;

    do {
        printf("\nDoubly Linked List Operations:\n");
        printf("1. Create a doubly linked list\n");
        printf("2. Insert a new node to the left of a node\n");
        printf("3. Delete a node based on a specific value\n");
        printf("4. Display the contents of the list\n");
        printf("5. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                {
                    int numNodes;
                    printf("Enter the number of nodes to create: ");
                    scanf("%d", &numNodes);

                    for (int i = 1; i <= numNodes; ++i) {
                        printf("Enter data for node %d: ", i);
                        scanf("%d", &data);
                        struct Node* newNode = createNode(data);

                        if (head == NULL) {
                            head = newNode;
                        } else {
                            struct Node* current = head;
                            while (current->next != NULL) {
                                current = current->next;
                            }
                            current->next = newNode;
                            newNode->prev = current;
                        }
                    }
                    break;
                }

            case 2:
                {
                    int targetData;
                    printf("Enter the data value of the node to the left of which you want to insert: ");
                    scanf("%d", &targetData);

                    struct Node* target = head;
                    while (target != NULL && target->data != targetData) {
                        target = target->next;
                    }

                    if (target == NULL) {
                        printf("Node with data %d not found.\n", targetData);
                    } else {
                        printf("Enter the data for the new node: ");
                        scanf("%d", &data);
                        insertLeft(&head, target, data);
                        printf("Node inserted successfully to the left of the node with data %d.\n",
targetData);
                    }
                    break;
                }

            case 3:
                {
                    printf("Enter the data value of the node to delete: ");
                    scanf("%d", &key);
                    deleteNode(&head, key);
                    break;
                }

            case 4:
                display(head);
                break;

            case 5:
                printf("Exiting program.\n");
                break;

            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }

    } while (choice != 5);

    return 0;
}
```

OUTPUT:



```
Doubly Linked List Operations:
1. Create a doubly linked list
2. Insert a new node to the left of a node
3. Delete a node based on a specific value
4. Display the contents of the list
5. Exit
Enter your choice: 1
Enter the number of nodes to create: 3
Enter data for node 1: 20
Enter data for node 2: 21
Enter data for node 3: 22

Doubly Linked List Operations:
1. Create a doubly linked list
2. Insert a new node to the left of a node
3. Delete a node based on a specific value
4. Display the contents of the list
5. Exit
Enter your choice: 2
Enter the data value of the node to the left of which you want to insert: 21
Enter the data for the new node: 21
Node inserted successfully to the left of the node with data 21.

Doubly Linked List Operations:
1. Create a doubly linked list
2. Insert a new node to the left of a node
3. Delete a node based on a specific value
4. Display the contents of the list
5. Exit
Enter your choice: 3
Enter the data value of the node to delete: 20
Node with value 20 deleted successfully.

Doubly Linked List Operations:
1. Create a doubly linked list
2. Insert a new node to the left of a node
3. Delete a node based on a specific value
4. Display the contents of the list
```



```
5. Exit
Enter your choice: 4
Doubly Linked List: 21 <-> 21 <-> 22 <-> NULL

Doubly Linked List Operations:
1. Create a doubly linked list
2. Insert a new node to the left of a node
3. Delete a node based on a specific value
4. Display the contents of the list
5. Exit
Enter your choice: 5
Exiting program.
```